

# A Distributed Enumeration Algorithm and Applications to All Pairs Shortest Paths, Diameter...

Y. Métivier, J.M. Robson, and A. Zemmari

Université de Bordeaux, LaBRI, UMR CNRS 5800  
351 cours de la Libération, 33405 Talence, France  
{metivier, robson, zemmari}@labri.fr

**Abstract.** We consider the standard message passing model; we assume the system is fully synchronous: all processes start at the same time and time proceeds in synchronised rounds. In each round each vertex can transmit a different message of size  $O(1)$  to each of its neighbours. This paper proposes and analyses a distributed enumeration algorithm of vertices of a graph having a distinguished vertex which satisfies that two vertices with consecutive numbers are at distance at most 3. We prove that its time complexity is  $O(n)$  where  $n$  is the number of vertices of the graph. Furthermore, the size of each message is  $O(1)$  thus its bit complexity is also  $O(n)$ . We provide some links between this enumeration and Hamiltonian graphs from which we deduce that this enumeration is optimal in the sense that there does not exist an enumeration which satisfies that two vertices with consecutive numbers are at distance at most 2.

We deduce from this enumeration algorithms which compute all pairs shortest paths and the diameter with a time complexity and a bit complexity equal to  $O(n)$ . This improves the best known distributed algorithms (under the same hypotheses) for computing all pairs shortest paths or the diameter presented in [PRT12,HW12] having a time complexity equal to  $O(n)$  and which use messages of size  $O(\log n)$  bits.

**Keywords:** Distributed Algorithm, Biconnectivity, Bit Complexity, Cut-Edge, Cut-Vertex, Diameter, Girth, Hamiltonian Graph.

## 1 Introduction

### 1.1 The problem

In this paper we consider the all pairs shortest paths problem in a distributed network. We assume that there exists a distinguished vertex (called *Leader*) so that there exist distributed algorithms for solving it. We are interested in optimal solutions in time and in number of bits for this problem.

Distributed algorithms for solving the all pairs shortest paths problem find extensive use in communication networks and thus in distributed computing. For example, the shortest path between a source and a destination is considered as the most economic. Finally, many routing schemes use shortest paths and designing such a scheme consists of computing the shortest routes and storing information on vertices for routing messages ([Pel00], p. 105).

The solution presented in this paper is based on a non trivial distributed enumeration algorithm which satisfies that two vertices with consecutive numbers are at distance at most 3.

### 1.2 The Model

**The Network.** We consider the standard message passing model for distributed computing. The communication model consists of a point-to-point communication network described by a connected graph  $G = (V(G), E(G))$  ( $= (V, E)$  for short) where the vertices  $V$  represent network processes and the edges

$E$  represent bidirectional communication channels. Processes communicate by message passing: a process sends a message to another by depositing the message in the corresponding channel. In the sequel, we consider only connected graphs. We assume the system is fully synchronous, namely, all processes start at the same time and time proceeds in synchronised rounds.

**Time Complexity.** A round (cycle) of each process is composed of the following three steps: 1. Send messages to (some of) the neighbours, 2. Receive messages from (some of) the neighbours, 3. Perform some local computation. As usual the time complexity is the number of rounds needed until every vertex has completed its computation.

**Bit Complexity.** We follow the definition given in [KOSS06]. By definition, in a bit round each vertex can send/receive at most 1 bit from each of its neighbours. The bit complexity of algorithm  $\mathcal{A}$  is the number of bit rounds to complete algorithm  $\mathcal{A}$ .

*Remark 1.* A round of an algorithm consists of 1 or more bit rounds. The bit complexity of a distributed algorithm is an upper bound on the total number of bits exchanged per channel during its execution. It is also an upper bound on its time complexity.

If we consider a distributed algorithm having messages of size  $O(1)$  (and this is the case in this paper) then the time complexity and the bit complexity are equal modulo a multiplicative constant.

The bit complexity is considered as a finer measure of communication complexity and it has been studied for breaking symmetry or for colouring in [BNN90,BMW94] or in [KOSS06,DMR08]. Dinitz et al. explain in [DMR08] that it may be viewed as a natural extension of communication complexity (introduced by Yao [Yao79]) to the analysis of tasks in a distributed setting. An introduction to this area can be found in Kushilevitz and Nisan [KN99].

**Network and Processes Knowledge.** The network  $G = (V, E)$  is anonymous: unique identities are not available to distinguish the processes. We only assume that there is an elected (a distinguished) vertex denoted *Leader*. We do not assume any global knowledge of the network, not even its size or an upper bound on its size. The processes do not require any position or distance information. Each process knows from which channel it receives or to which channel it sends a message, thus one supposes that the network is represented by a connected graph with a port numbering function defined as follows (where  $I_G(u)$  denotes the set of edges of  $G$  incident to  $u$ ):

**Definition 1.** Given a graph  $G = (V, E)$ , a port numbering function  $\delta$  is a set of local functions  $\{\delta_u \mid u \in V\}$  such that for each vertex  $u \in V$ ,  $\delta_u$  is a bijection between  $I_G(u)$  and  $[1, \deg_G(u)]$ .

**All Pairs Shortest Paths, Diameter, Girth, Cut-Edge and Cut-Vertex.** We follow definitions given in [Ros00]. A walk in a graph  $G = (V, E)$  is a finite alternating sequence of vertices and edges, beginning and ending with a vertex and where each edge is incident with the vertices immediately preceding and following it. A trail is a walk in which no edge occurs more than once. A path is a trail in which all of its vertices are different, except that the initial and final vertices may be the same. A walk with at least 3 vertices in which the first and last vertices are the same but all other vertices are distinct is called a cycle.

Let  $G = (V, E)$  be a connected graph, let  $u, v \in V$ . The distance between  $u$  and  $v$  in  $G$ , denoted  $dist_G(u, v)$ , is the length of a shortest path between  $u$  and  $v$  in  $G$ .

Given a vertex  $v$  of a connected graph, the eccentricity of  $v$  is the greatest distance from  $v$  to another vertex.

The all pairs shortest paths (APSP for short) problem in  $G$  is to compute the length of shortest paths between any pair of vertices in  $G$ .

The diameter of  $G$ , denoted  $D(G)$ , is the maximum distance between any two vertices of  $G$ , i.e.,  $D(G) = \max\{\text{dist}_G(u, v) \mid u, v \in V\}$ .

The girth of a graph  $G$  is the length of a shortest cycle of  $G$ .

A cut-vertex is a vertex whose removal increases the number of connected components.

A cut-edge is an edge whose removal increases the number of connected components.

We use trees and we follow the presentation given in [CLRS09]. A tree is a connected acyclic graph. A rooted tree is a tree in which one of the vertices is distinguished from the others (called *Leader* in this work). A spanning-tree of a connected graph  $G = (V, E)$  is a tree  $T = (V, E')$  such that  $E' \subseteq E$ .

### 1.3 Our Contribution

We present a distributed enumeration algorithm, denoted *DEA*, which assigns to each vertex of a graph  $G$  of size  $n$  having a distinguished vertex, denoted *Leader*, a unique integer of  $\{1, 2, \dots, n\}$  such that the distance between any two vertices having two consecutive numbers is at most 3. This algorithm uses messages of size  $O(1)$  and has a time complexity equal to  $O(n)$ .

The steps of Algorithm *DEA* are:

1. computation of a Breadth-First-Search (BFS) spanning-tree of  $G$  whose root is *Leader*;
2. enumeration of the vertices with respect to a special traversal of the BFS spanning-tree.

This enumeration enables the initialisation of anonymous waves, with respect to the enumeration order, i.e., the first wave is initialised by the vertex numbered 1, the second wave by the vertex numbered 2, etc. Anonymous waves reach vertices with respect to the enumeration order (i.e., the wave initialised by the vertex numbered  $i$  reaches any vertex after the wave initialised by the vertex numbered  $i - 1$  and before the wave initialised by the vertex numbered  $i + 1$ ) and thus are implicitly identified. This fact allows each vertex to compute its distance to any vertex without the computation and the use of the distance itself but by inference from the time; in this way all pairs shortest paths are obtained in time  $O(n)$ , each message having a constant size so that the bit complexity is  $O(n)$ .

We deduce also a distributed algorithm for graph diameter with a bit complexity and a time complexity equal to  $O(n)$ .

Frischknecht et al. proved ([FHW12], Theorem 5.1) that: “For any  $n \geq 10$  and  $B \geq 1$  and sufficiently small  $\epsilon$  any distributed randomized  $\epsilon$ -error algorithm  $A$  that computes the exact diameter of a graph requires at least  $\Omega(n/B)$  time for some  $n$ -node graph even when the diameter is at most 5,” where  $B$  is the size of messages.

From this result we deduce that the bit complexity of our algorithm is optimal and the time complexity is also optimal for messages of size  $O(1)$ .

In the remainder of this work, we explain how the enumeration algorithm can be applied for computing the girth, cut-edges, cut-vertices or for recognising biconnected graphs.

*Remark 2.* We may wonder whether it is possible to obtain an enumeration of vertices such that the distance between any two vertices having two consecutive numbers is at most 2. We explain in the next section why the answer is negative. It indicates that in some certain sense our enumeration is optimal.

### 1.4 Related Work: Comparisons and Comments

**Enumeration Algorithm.** The enumeration of vertices of a connected graph such that two consecutive vertices of the enumeration are at distance at most 3 is also presented in [Sek71]; this paper presents a sequential algorithm for computing such an enumeration.

Let  $G$  be a graph. A Hamiltonian path in  $G$  is a path that includes all the vertices of  $G$ .

We recall that the cube of a graph  $G$ , denoted  $G^3$ , is the graph with the set of vertices of  $G$  in which there is an edge between two vertices  $u$  and  $v$  if the distance between  $u$  and  $v$  in  $G$  is at most 3. It was noticed by C. Gaville [Gav14] that the existence of such an enumeration is equivalent to the fact that the cube of a connected graph  $G$  contains a Hamiltonian path

A cycle containing all vertices of  $G$  is called a Hamiltonian cycle of  $G$ , and  $G$  is called a Hamiltonian graph. From our enumeration result we deduce a well known result [CK69]:

**Theorem 1.** *If  $G$  is a connected graph then  $G^3$  is a Hamiltonian graph.*

As for the cube of a graph, the square of a graph  $G$ , denoted  $G^2$ , is the graph with the set of vertices of  $G$  in which there is an edge between two vertices  $u$  and  $v$  if the distance between  $u$  and  $v$  in  $G$  is at most 2.

The previous theorem is no longer true for the square of a tree as indicated by the next theorem. Let  $K_{1,3}$  be the tree with one internal vertex and three leaves. Let  $S(K_{1,3})$  be the subdivision of  $K_{1,3}$  formed by inserting a vertex of degree two on each edge of  $K_{1,3}$ . A graph  $G' = (V', E')$  is called a subgraph of a graph  $G = (V, E)$  if  $V' \subseteq V$ ,  $E' \subseteq E$  and  $V'$  contains all the endpoints of the edges in  $E'$ . Regarding the characterisation of trees with Hamiltonian square, Harary and Schwenk [HS71] proved that:

**Theorem 2.** *Let  $T$  be a tree with at least 3 vertices.  $T^2$  is a Hamiltonian graph if and only if  $T$  does not contain  $S(K_{1,3})$  as a subgraph.*

In fact for our work, a priori, we only need a Hamiltonian path. Remark 2 is a direct consequence of the following result. In [RR11], it is proved that the square of a tree  $T$  has a Hamiltonian path if and only if  $T$  is a horsetail. The definition of a horsetail is rather technical thus we do not provide it; in our context the important fact is that the family of trees which are not horsetails is infinite.

**All Shortest paths, Diameter, Girth...** The computation of all pairs shortest paths, of the diameter or of the girth is the subject of many studies. Very complete recent surveys on these questions and on associated results can be found in [FHW12,HW12,PRT12,LPS13,Nan14].

Known results depend on the size of the messages, denoted  $B$  in the sequel, that a vertex can transmit to its neighbours. Furthermore it depends also on whether algorithms compute exact values or approximations (approximations enable in some cases an improvement in the running time). Among the most recent results one can cite: [WW10,ABC11,RT11,RW11,RW12,FHW12] [HW12,PRT12], [LPS13,Nan14].

Frischknecht et al. [FHW12] established an  $\Omega(n/B)$  lower bound for the number of communication rounds needed for computing the diameter of a graph (they use a non-trivial technique of transferring lower bounds from communication complexity and graph-constructions). Thus the challenge for the computation of all pairs shortest paths or of the diameter in linear time (in this context) concerns the size of messages. Almeida et al. present in [ABC11] an algorithm with a time complexity  $O(D)$  (where  $D$  is the diameter of the graph) with large messages:  $B = O(n \log n)$ . The best known distributed algorithms for computing the diameter with  $B = O(\log n)$  are presented in [PRT12,HW12]. Both assume that the size  $n$  of the graph is known and each vertex has a unique identifier from  $\{1, \dots, n\}$ . In both cases, algorithms compute BFS spanning-trees rooted at each vertex and compute distances between any two vertices. The time complexity of both algorithms is  $O(n)$ . The key point is that there is no collision between messages of different BFS spanning-trees construction processes: at any time a vertex is active for the construction of at most one BFS spanning-tree. Messages enable the computation of distances between vertices so that the size of messages is  $O(\log n)$  and the bit complexity of both algorithms is  $O(n \log n)$ .

*Remark 3.* Our initial knowledge and hypotheses on graphs are equivalent to the initial knowledge and initial hypotheses on graphs in [PRT12,HW12] in the sense that one can be obtained from the other in a linear time with a linear bit complexity.

	Time	Message size (number of bits)	bit complexity
Almeida et al. [ABC11]	$O(D)$	$O(n \log n)$	$O(Dn \log n)$
Holzer and Wattenhofer [HW12]	$O(n)$	$O(\log n)$	$O(n \log n)$
Peleg et al. [PRT12]	$O(n)$	$O(\log n)$	$O(n \log n)$
This paper	$O(n)$	$O(1)$	$O(n)$

This table summarises the comparison between the complexities of various diameter algorithms and the complexities of the diameter algorithm presented in this paper.

General considerations and results concerning cut-edges, cut-vertices and biconnectivity are presented in [PT11, Cha98, Hoh90]. Thurimella [Thu97] proved that the diameter is a more precise parameter for the time complexity of finding cut-edges, cut-vertices or deciding the biconnectivity, more precisely, Thurimella obtained time complexity  $O(D + \sqrt{n} \log^* n)$  for these problems on a graph  $G$  where  $D$  is the diameter of  $G$ .

This paper is organised as follows. Section 2 presents a distributed enumeration algorithm, denoted *DEA*, and proves that its time complexity and its bit complexity are linear. Section 3 applies Algorithm *DEA* to all pairs shortest paths. Section 4 gives an immediate application for computing the diameter. Section 5 explains how anonymous waves enable the computation of the girth, cut-edges and cut-vertices with a linear time complexity and a linear bit complexity.

## 2 A Distributed Enumeration Algorithm

This section describes the steps of Algorithm *DEA* which enumerates vertices of a given graph  $G$  having a distinguished vertex, denoted *Leader*,:

1. computation of a BFS spanning-tree of  $G$  whose root is *Leader*;
2. enumeration of the vertices with respect to a special traversal of the BFS spanning-tree.

Consider a vertex  $v$  in a rooted tree with root *Leader*. The length of the unique path from *Leader* to  $v$  is the level of  $v$ . Any node  $w$  on the unique path from *Leader* to  $v$  is called an ancestor of  $v$ . If the last edge on the unique path from *Leader* to a vertex  $v$  is  $\{w, v\}$  then  $w$  is the parent of  $v$  and  $v$  is a child of  $w$ . A leaf is a vertex with no child. Two vertices  $v$  and  $w$  are brothers if they have the same parent. We consider ordered trees (also called plane trees), meaning that in the definition above a total order is assigned to each set of children of each vertex (in our case, the total order is the order induced by the port numbering). Thus if we consider a vertex  $v$  having  $k$  children we can speak of the first child, of the second child etc. If  $v$  and  $w$  are brothers, let  $u$  be their parent;  $w$  is said to be the next brother of  $v$  if it is the next successor of  $v$  with respect to the total order assigned to the children of  $u$ .

### 2.1 Computing the Breadth-First-Search Spanning-Tree Rooted at Leader

The first step of Algorithm *DEA* computes a BFS spanning-tree. Starting from *Leader*, the spanning-tree, denoted *BFS-ST*, is computed level by level by the well-known procedure BFS (see [Pel00] p. 50).

Initially each vertex is in the state *waiting*, in detail:

1. *Leader* sends the signal *Start* to all its neighbours;

2. any vertex in state *waiting* receiving a *Start* signal from one or more neighbours at time  $t$  does:
  - (a) chooses as its parent the first such neighbour (in the order of enumeration of its ports, for example);
  - (b) at time  $t + 1$  sends *Accept* to its parent and *Reject* to the other neighbours that sent *Start*;
  - (c) at time  $t + 1$  sends *Start* to all its other neighbours;
  - (d) at time  $t + 2$  notes as its child any neighbours sending *Accept* and sends *Reject* to all other neighbours that sent *Start* at time  $t + 1$ ;
3. as soon as a non-leader vertex has carried out step 2.(c) and received *OK* from all its children (a leaf has no child), it sends *OK* to its parent;
4. as soon as Leader has sent *Start* and received *OK* from all its children (that is all its neighbours), it knows the breadth-first-search tree computation is complete.

**Claim 1** *Let  $G$  be a graph having  $n$  vertices and a distinguished vertex Leader. The procedure BFS computes a BFS spanning-tree of  $G$  in time  $O(n)$ . Its bit complexity is also  $O(n)$ .*

We recall that edges of  $G$  that do not appear in the BFS spanning-tree connect vertices either with the same level or with levels which differ by at most 1.

## 2.2 A Distributed Enumeration Algorithm

Once Leader knows that the BFS spanning-tree computation is complete, it starts a phase in which each vertex in turn (in an order to be described later) starts a wave propagation which traverses the whole graph. Thanks to the properties of the order, each vertex can calculate its distance from any other vertex. To define the order in which the waves are started, we define a traversal of the BFS spanning-tree and a numbering of vertices.

First we define a tree traversal, denoted *Trav*, used to visit and to number systematically each vertex.

This tree traversal and the associated vertex numbering may be defined iteratively as follows. We add a loop on each leaf of the BFS spanning-tree and vertices are visited twice in a Depth-First-Search (DFS) traversal (a leaf is visited on arriving and by following the loop).

The traversal *Trav* is defined by:

- if it is the first visit to a vertex then go to the first child of the vertex if it has a child; if it has no child (i.e., it is a leaf) go from the leaf to itself;
- if it is the second visit to a vertex go to the next brother of the vertex, if it has a next brother; if it has no next brother go to the parent of the vertex if it has a parent else stop since it is the root of the BFS spanning-tree and the traversal is finished.

Thus, a visit to  $v$  is immediately followed by a visit to a child or to a brother or to the parent of  $v$  or to  $v$  itself (if  $v$  is a leaf).

**Claim 2** *Each vertex is visited twice.*

Let  $v$  be a vertex,  $\nu_v^{(1)}$  (resp.  $\nu_v^{(2)}$ ) denotes the number of vertices visited before the first visit to  $v$  (resp. before the second visit to  $v$ ).

By an induction on the level of vertices:

**Lemma 1.** *Let  $v$  be a vertex,  $\nu_v^{(1)}$  is even if and only if the level of  $v$  is even;  $\nu_v^{(2)}$  is even if and only if the level of  $v$  is odd.*

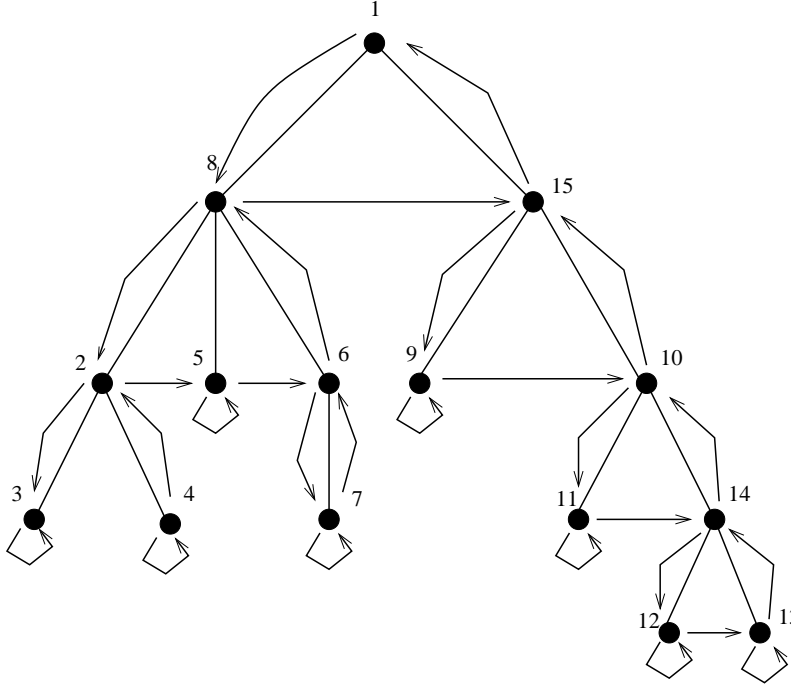
From the previous lemma:

**Corollary 1.** *Let  $v$  be a vertex. For each run of *Trav*, if  $\nu_v^{(1)}$  is odd (resp. even) then  $\nu_v^{(2)}$  is even (resp. odd).*

The number of a vertex  $v$  is obtained by computing the number of visited vertices during the tree traversal before the first or the second visit to  $v$ . More precisely:

**Definition 2.** *The vertex numbered  $k$  is the  $k^{\text{th}}$  visited vertex such that an even number of vertices have been visited before it; it is denoted  $v_k$ .*

An example of a run of *Trav* and the numbering of vertices is given in Fig. 1.



**Fig. 1.** An example of a run of *Trav* with the associated numbering of the vertices.

By induction on the level of a vertex, we have:

**Lemma 2.** *Let  $v$  be the vertex numbered  $k$ . If  $\nu_v^{(1)}$  is even then  $k = \nu_v^{(1)}/2 + 1$  else  $k = \nu_v^{(2)}/2 + 1$ .*

Now, we can state the fundamental property of the numbering we use later:

**Lemma 3.** *Let  $G$  be a connected graph. We consider the numbering of vertices of  $G$  after a run of *Trav*. The distance between  $v_i$  and  $v_{i+1}$  is at most 3 in the BFS spanning-tree and thus in  $G$ .*

*Proof.* Two vertices having consecutive numbers are separated by two steps in the traversal of the tree.

Furthermore a visit to  $v$  is immediately followed by a visit to a child or a brother or the parent of  $v$  or  $v$  itself (if  $v$  is a leaf). Therefore the distance between a vertex  $v$  and the vertex  $w$  reached after two steps is at most 3. The result follows.

The numbering of the vertices can be achieved by combining steps of the traversal and sending messages 1 (using unary representation of the numbers) and *End*. More precisely:

1. Leader sends messages 1 and *End* on two successive steps to its first child; the number of Leader is 1;



2. a vertex receiving 1 sends 1 to its successor with respect to the *Trav* traversal (i.e., to a child, a brother, the parent or to itself (for a leaf)) on the following step;
3. a vertex receiving *End* sends 1 and *End* to its successor with respect to the tree traversal on the two following steps;
4. a vertex  $v$  receives 1 from two predecessors in traversal order; let  $p_1$  and  $p_2$  be the numbers of 1 received by  $v$  from these two predecessors for this traversal; one is even and the other is odd. Assume  $p_1$  is even; then the number of  $v$  is  $p_1/2 + 1$ ;
5. as soon as Leader has received *End* from its last child it knows the numbering is complete.

As in previous sections:

**Claim 3** *Let  $G$  be a connected graph having  $n$  vertices. Enumeration of the vertices of  $G$  has a time complexity and a bit complexity equal to  $O(n)$ .*

### 3 All Pairs Shortest Paths

The computation of All Pairs Shortest paths needs, first, the computation by each vertex of its distance to Leader.

#### 3.1 Calculating Distance from Leader to each Vertex

Once Leader knows that the tree computation is complete, it starts a process in which each other vertex learns its distance from Leader. Leader sends a signal to each of its neighbours telling it that its distance is 1 and every vertex  $v$  sends to all its children a message giving the child's distance as 1 more than that of  $v$ . For simplicity, we describe a method of achieving this using unary representation of the distances. Thus the distance computation is obtained by the procedure *Dist-Cal* defined by:

1. Leader sends messages 1 and *End* to each child on two successive steps;
2. a vertex receiving 1 sends 1 to each of its children on the following step;
3. a vertex receiving *End* sends 1 and *End* to its children on the two following steps;
4. the distance of a vertex from Leader is the number of 1s received;
5. as soon as a non-leader vertex has sent *End* to and received OK from all its children (including the case of a leaf which has no child), it sends OK to its parent;
6. as soon as Leader has received OK from all its children (that is all its neighbours), it knows the distance computation is complete.

**Claim 4** *Let  $G$  be a graph having  $n$  vertices and a distinguished vertex Leader. The procedure *Dist-Cal* enables each vertex to know its distance to Leader; the time complexity and the bit complexity required are  $O(n)$ .*

*Remark 4.* As we consider a BFS spanning-tree, the level of a vertex is its distance to the root.

#### 3.2 All Pairs Shortest Paths

Once Leader knows that the enumeration of the vertices is completed and each node knows its distance to Leader, it starts a phase in which every vertex in the order of the enumeration starts an anonymous wave propagation for distance calculation. The propagation of a wave follows these rules:

1. the source of the wave sends *wave* once to all its neighbours;
2. a vertex  $v$  receiving *wave* at time  $t$  from one or more neighbours sends *wave* at time  $t + 1$  to all other neighbours;



3.  $v$  ignores any *wave* signal received at time  $t + 1$  (from any neighbour at the same distance from the wave's source).

Based on this mechanism, we give the algorithm *APSP* for the all pairs shortest paths calculation:

1. Leader starts a wave at time  $t_1$ ;
2. each vertex computes  $t_1$  when receiving the first wave;
3. each vertex  $v_i$  starts a wave at time  $t_i = t_1 + 5(i - 1)$ ;
4. each vertex  $v_i$  computes its distance to any vertex  $v_j$  when it receives the  $j^{th}$  wave;
5. each vertex  $v$  computes the maximal distance to any vertex when no new wave arrives within eight steps after the last one.

First:

**Claim 5** *Let  $v$  be a vertex. If  $v$  starts a wave following rules above then each vertex  $w$  receives wave signals at times  $d$  and (possibly)  $d + 1$  after the start of the propagation where  $d$  is the distance between  $v$  and  $w$ .*

**Lemma 4.** *The waves start by two consecutive vertices (with respect to the enumeration order) won't collide with each other.*

*Proof.* Let  $t_1$  be the time at which *Leader* starts its wave. Every vertex knows its distance to *Leader* and thus can compute  $t_1$  as soon as it receives the signal *wave* for the first time.

Next, every vertex  $v_i$  starts a new wave (denoted  $w_i$ ) at time  $t_i = t_1 + 5(i - 1)$ .

Since the wave  $w_{i+1}$  starts 5 rounds after the wave  $w_i$  and the distance between  $v_i$  and  $v_{i+1}$  is at most 3, these two waves arrive at any vertex  $v$  separately and in the order  $w_i$  followed by  $w_{i+1}$  and at distance at least 2.

**Lemma 5.** *Each vertex can compute its exact distance to each other vertex.*

*Proof.* Let  $\tau_1, \dots, \tau_n$  denote the times of arrival of the  $n$  waves at  $v$ . We have:  $\tau_1 = t_1 + d(v, \text{root})$ . Now,  $\tau_i = t_i + d(v, v_i)$  thus  $d(v, v_i) = \tau_i - t_i = \tau_i - t_1 - 5(i - 1)$ , and  $v$  can compute its distance from  $v_i$ .

Hence when no new wave arrives at a vertex  $v$  (i.e., no new wave arrives within eight steps after the last one),  $v$  knows its exact distance from each other vertex.

Finally:

**Theorem 3.** *Let  $G$  be a graph having  $n$  vertices and a distinguished vertex. There exists a synchronous distributed algorithm which computes APSP of  $G$  in  $O(n)$  rounds with a bit complexity equal to  $O(n)$ .*

## 4 Computing the Diameter

This section indicates how to compute the diameter of  $G$  by centralising the maximum distance and broadcasting the result.

**Theorem 4.** *Let  $G$  be a graph having  $n$  vertices and a distinguished vertex. There exists a synchronous distributed algorithm which computes the diameter of  $G$  in  $O(n)$  rounds with a bit complexity equal to  $O(n)$ . Furthermore each vertex knows the value of the diameter at the end of the algorithm.*

*Proof.* The eccentricities calculated in the previous section are now sent up via the BFS spanning-tree to its root.

For a vertex  $v$ , we write  $m_v$  for the eccentricity of  $v$ .

Each vertex  $v$  except the root will send to its parent the maximum distance, denoted  $M_v$ , from any vertex in the subtree rooted at  $v$ ; this value will be sent in  $M_v + 2$  consecutive rounds in the form of one

$max$  signal followed by  $M_v$  1 signals and one  $endmax$  signal. A leaf can start this process as soon as it knows its own maximum distance  $m_v$  since  $M_v = m_v$ .

A non-leaf vertex  $v$  will wait until it has received the  $max$  signal from each of its children  $w$ . It then sends the  $max$  signal to its parent and continues to send 1 signals until it has received the  $endmax$  signal from each child. It now knows the  $M_w$  for each of its subtrees and  $m_v$  and so can compute  $M_v$  and send the required number of extra 1 signals to its parent. Note that the number of 1 signals already sent at this point is at most  $M_w$  for any child  $w$  which sent the last  $endmax$  and so cannot be greater than  $M_v$ .

In this way the root knows the global maximum after diameter, and thus at most a linear, number of rounds.

Finally, the root sends the global maximum (it suffices to use the same unary format as previously) to each of its children who transmit it to each of their children etc. Again this takes a linear number of rounds.

Finally, the diameter is obtained by the following steps:

1. Breadth-First-Search Tree Computation initiated by Leader;
2. Numbering of vertices;
3. Calculating distance between Leader and each vertex;
4. Waves initiation and all pairs shortest paths calculation;
5. Centralisation of the maximum distance and broadcast of the diameter.

## 5 Other Applications of the Numbering of Vertices

We illustrate the power of the waves initiated by vertices with respect to the numbering of vertices for computing girth and for the determination of cut-edges and cut-vertices.

### 5.1 Computing the Girth

**Theorem 5.** *Let  $G$  be a graph having  $n$  vertices and a distinguished vertex. The girth of  $G$  can be computed by a distributed algorithm and known by each vertex with a time and a bit complexity equal to  $O(n)$ .*

*Proof.* As for the computation of the diameter, once Leader knows that the enumeration of the vertices is complete it starts a phase in which every vertex in the order of the enumeration starts an anonymous wave propagation.

Let  $v$  be a vertex. If  $v$  receives the signal *wave* from at least two neighbours at time  $d$  then it concludes that it belongs to a cycle of length  $2d$ . If  $v$  receives the signal *wave* from a neighbour at time  $d$  and the signal *wave* at time  $d + 1$  from another neighbour then it concludes that it belongs to a cycle of length  $2d + 1$ .

If  $v$  belongs to a cycle then there is at least one vertex  $u$  of this cycle such that  $u$  starts a wave and this wave will reach  $v$  simultaneously by two different edges incident to  $v$  or will reach  $v$  by an edge incident to  $v$  at time  $d$  and by another edge incident to  $v$  at time  $d + 1$ . Thus the length of this cycle will be calculated by  $v$ .

When  $v$  knows that no new wave will arrive (i.e., no new wave arrives within eight steps after the last one) it computes the minimal length of cycles to which it belongs, denoted  $c_v$ .

If  $v$  belongs to no cycle then, by convention the length is 0.

Now, each vertex sends to Leader  $(n - c_v)$  by following the same procedure as in the previous section. It has found  $n$  by counting the number of waves.

Finally, as for the diameter, Leader centralises this value and it deduces and transmits the girth.

## 5.2 Computing Cut-Edges

Let  $G$  be a connected graph; an edge is said to be a cut-edge if its deletion disconnects  $G$ .

**Theorem 6.** *Let  $G$  be a connected graph having  $n$  vertices and a distinguished vertex. Cut-edges can be determined by a distributed algorithm and known by endpoints of cut-edges in  $O(n)$  rounds with a bit complexity equal to  $O(n)$ .*

*Proof.* As for the computation of the diameter, once Leader knows that the enumeration of the vertices is complete it starts a phase in which every vertex in the order of the enumeration starts a wave propagation.

Let  $v$  be a vertex. Let  $e$  be an edge incident to  $v$ . The theorem is a direct consequence of the following fact:

Edge  $e$  is a cut-edge if and only if, whenever  $v$  receives the signal *wave* through  $e$  at time  $t$ , it does not receive it through another edge at time  $t$  or  $t + 1$ .

## 5.3 Computing Cut-Vertices, Recognising Biconnected Graphs

Let  $G$  be a connected graph. Let  $v$  be a vertex of  $G$ . The vertex  $v$  is a cut-vertex if removing  $v$  and edges incident to  $v$  disconnects  $G$ .

**Theorem 7.** *Let  $G$  be a connected graph with  $n$  vertices and a distinguished vertex. Cut-vertices of  $G$  can be determined by a distributed algorithm in  $O(n)$  rounds with a bit complexity equal to  $O(n)$ .*

*Proof.* As previously, once Leader knows that the enumeration of the vertices is complete it starts a phase in which every vertex in the order of the enumeration starts a wave propagation.

Let  $u$  be a vertex having at least 2 neighbours. We define the relation  $R_u$  as follows: two vertices  $v_1$  and  $v_2$  are related modulo the relation  $R_u$  if the vertex  $u$  receives the signal *wave* from  $v_1$  at time  $t$  and the signal *wave* from  $v_2$  at time  $t$  or at time  $t + 1$  for some time  $t$ .

The theorem is a direct consequence of the following fact: the vertex  $u$  is not a cut-vertex if and only if every pair of neighbours of  $u$  is related modulo the transitive closure of  $R_u$ .

This theorem has the following corollary:

**Corollary 2.** *Let  $G$  be a connected graph with  $n$  vertices and a distinguished vertex. To know whether  $G$  is biconnected can be determined by a distributed algorithm in  $O(n)$  rounds with a bit complexity equal to  $O(n)$ .*

*Proof.* The result follows from the fact that a graph is biconnected if and only if it has no cut-vertex.

## 6 Conclusion

This work has been motivated by the distributed computation of all pairs shortest paths and the diameter with short messages. We introduce a distributed enumeration algorithm which uses messages of size  $O(1)$ . From this enumeration we deduce algorithms for the computation of all pairs shortest paths and for the diameter which improve known results. Finally, this enumeration algorithm provides another proof of the fact that the cube of a tree is a Hamiltonian graph and computes a Hamiltonian cycle of the cube with a time complexity and a bit complexity equal to  $O(n)$ .

## References

- ABC11. P. S. Almeida, C. Baquero, and A. Cunha. Fast distributed computation of distances in networks. *CoRR*, abs/1111.6087, 2011.
- BMW94. H. L. Bodlaender, S. Moran, and M. K. Warmuth. The distributed bit complexity of the ring: from the anonymous case to the non-anonymous case. *Inf. and comput.*, 114(2):34–50, 1994.
- BNNN90. A. Bar-Noy, J. Naor, and M. Naor. One-bit algorithms. *Distributed Computing*, 4:3–8, 1990.
- Cha98. P. Chaudhuri. An optimal distributed algorithm for finding articulation points in a network. *Computer Communications*, 21(18):1707–1715, 1998.
- CK69. G. Chartrand and S.F. Kapoor. The cube of every connected graph is 1-hamiltonian. *J. Res. Nat. Bur. Standards Sect B.*, 73B:47–48, 1969.
- CLRS09. Th. H. Cormen, Ch. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009.
- DMR08. Y. Dinitz, S. Moran, and S. Rajsbaum. Bit complexity of breaking and achieving symmetry in chains and rings. *Journal of the ACM*, 55(1), 2008.
- FHW12. S. Frischknecht, S. Holzer, and R. Wattenhofer. Networks cannot compute their diameter in sublinear time. In *SODA*, pages 1150–1162, 2012.
- Gav14. C. Gavaille. Private communication. 2014.
- Hoh90. W. Hohberg. How to find biconnected components in distributed networks. *J. Parallel Distrib. Comput.*, 9(4):374–386, 1990.
- HS71. F. Harary and A. Schwenk. Trees with hamiltonian square. *Mathematika*, 18:138–140, 1971.
- HW12. S. Holzer and R. Wattenhofer. Optimal distributed all pairs shortest paths and applications. In *PODC*, pages 355–364, 2012.
- KN99. E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge University Press, 1999.
- KOSS06. K. Kothapalli, M. Onus, C. Scheideler, and C. Schindelhauer. Distributed coloring in  $O(\sqrt{\log n})$  bit rounds. In *20th International Parallel and Distributed Processing Symposium (IPDPS 2006), Proceedings, 25-29 April 2006, Rhodes Island, Greece*. IEEE, 2006.
- LPS13. Ch. Lenzen and B. Patt-Shamir. Fast routing table construction using small messages: extended abstract. In *STOC*, pages 381–390, 2013.
- Nan14. D. Nanongkai. Distributed approximation algorithms for weighted shortest paths. In *STOC*, 2014.
- Pel00. D. Peleg. *Distributed computing - A Locality-sensitive approach*. SIAM Monographs on discrete mathematics and applications, 2000.
- PRT12. D. Peleg, L. Roditty, and E. Tal. Distributed algorithms for network diameter and girth. In *ICALP (2)*, pages 660–672, 2012.
- PT11. D. Pritchard and R. Thurimella. Fast computation of small cuts via cycle space sampling. *ACM Transactions on Algorithms*, 7(4):46, 2011.
- Ros00. *Handbook of Discrete and Combinatorial Mathematics*. CRC Press, 2000.
- RR11. J. Radoszewski and W. Rytter. Hamiltonian paths in the square of a tree. In *ISAAC*, pages 90–99, 2011.
- RT11. L. Roditty and R. Tov. Approximating the girth. In *SODA*, pages 1446–1454, 2011.
- RW11. L. Roditty and V. V. Williams. Minimum weight cycles and triangles: Equivalences and algorithms. In *FOCS*, pages 180–189, 2011.
- RW12. L. Roditty and V. V. Williams. Subquadratic time approximation algorithms for the girth. In *SODA*, pages 833–845, 2012.
- Sek71. M. Sekanina. On an algorithm for ordering of graphs. *Canad. Math. Bull.*, 14(2):221–224, 1971.
- Thu97. R. Thurimella. Sub-linear distributed algorithms for sparse certificates and biconnected components. *J. Algorithms*, 23(1):160–179, 1997.
- WW10. V. V. Williams and R. Williams. Subcubic equivalences between path, matrix and triangle problems. In *FOCS*, pages 645–654, 2010.
- Yao79. A. C. Yao. Some complexity questions related to distributed computing. In *Proceedings of the 11th ACM Symposium on Theory of computing (STOC)*, pages 209–213. ACM Press, 1979.